2D Triangulation Representation Using Stable Catalogs

Olivier Devillers, Abdelkrim Mebarki, Luca Castelli Aleardi

2D Triangulation Representation Using Stable Catalogs.*

Luca Castelli Aleardi† Olivier Devillers‡ Abdelkrim Mebarki§

Abstract

The problem of representing triangulations has been widely studied to obtain convenient encodings and space efficient data structures. In this paper we propose a new practical approach to reduce the amount of space needed to represent in main memory an arbitrary triangulation, while maintaining constant time for some basic queries. This work focuses on the connectivity information of the triangulation, rather than the geometry information (vertex coordinates), since the combinatorial data represents the main storage part of the structure. The main idea is to gather triangles into patches, to reduce the number of pointers by eliminating the internal pointers in the patches and reducing the multiple references to vertices. To accomplish this, we define stable catalogs of patches that are close under basic standard update operations such as insertion and deletion of vertices, and edge flips. We present some bounds and results concerning special catalogs, and some experimental results for the quadrilateral-triangle catalog.

1 Introduction

The triangulation is the basic data structure in a large spectrum of application domains, ranging from geometric applications, to finite element applications and interpolation schemes. This data structure has been widely studied from different points of view, and several schemes have been recently proposed for representing triangular meshes.

One can use Half-Edge-Based representation [11], in which the triangulation is perceived as a set of half-edges. Each half-edge is represented at least with one of its incident vertices, the opposite half-edge, and the previous (or the next) half-edge in the same incident face. Moreover, each vertex stores a reference to an incident half-edge, which yields a global storage cost of $19n$ references for a triangulation of n vertices.

In the Face-Based representation [4], the objects of the triangulation are faces (triangles). Each face maintains the three vertices references, and the three neighbors references. In addition, each vertex maintains a reference of an incident face, hence representing a triangulation with n vertices requires $13n$ references.

These structures allow an efficient navigation over the triangulation: standard adjacency queries, as visiting neighbors, and incident queries (testing incidence between faces, edges and vertices), are all supported in $O(1)$ time.

From the encoding point of view, many solutions have been developed for compression purpose, mainly for triangular meshes. In this case, the triangulation is just implicitly encoded (hence is not a data structure), and there does not exist an efficient way to access to the stored data, without the decompression of the whole code. From the information theory point of view, we know that representing an arbitrary planar triangulation requires $3.24bpv$ (bits per vertex), and a linear time optimal encoding has been recently introduced [13]. On the practical side, several efficient compression schemes have been proposed, achieving very interesting bit rates, especially in the case of regular meshes [2].

Beyond the usual representation schemes (Half-Edge based and Face based), some compact representations were proposed to reduce the memory cost.

In [10], a Vertex-Based representation is discussed. Each vertex handles a list of all of its adjacent vertices (the vertex stores the size of this list), resulting in $7n$ references to represent the whole triangulation. However, the internal structure no longer have an explicit representation of faces, and queries takes time proportional to the degree of an involved vertex.

In [3], a compact data structure for representing simplicial meshes is proposed, requiring in practice 5 *bytes/triangle*. The representation does admit an Edge-based or Vertex-based representation, providing basic update operations and standard local navigation between triangles (performing these operations takes $O(1)$ time for the case of meshes with bounded vertex degree). To gain in memory, difference vertex labels are used instead of real pointers, and a preprocessing step consisting of relabelling vertices, for reducing the differences, is needed (this approach takes advantage of properties of graphs with small separators).

Paper's contribution

Recently [6, 7], we have proposed an optimal way of representing a triangulation of n points using $3.24n$ bits,

*This work has been supported by the French "ACI Masse de Donnes" program, via the GeoComp project, http://www.lix.polythechnique.fr/ shaeffer/GeoComp

†INRIA & LIX, amturing@lix.polytechnique.fr

‡INRIA, Olivier.Devillers@sophia.inria.fr

§INRIA, Abdelkrim.Mebarki@sophia.inria.fr

with an additional storage which is asymptotically negligible (in the case of a triangulation of a topological sphere ; for the triangulation bounded by a polygon of arbitrary size the cost is 2.17 bits per triangle).

The idea is to gather triangles in tiny patches of size between $\frac{\log n}{12}$ and $\frac{\log n}{4}$, and to introduce a graph of patches to describe adjacency relations between them. Each patch is then represented by a reference to a catalog, consisting of all different tiny patches of size less than $\frac{\log n}{4}$. The whole size of all references to the catalog gives the dominant term of 3.24 *bpv*, while the representation of the graph of patches requires a negligible amount of space.

Unfortunately this negligible term is of the form $O\left(n\frac{\log\log n}{\log n}\right)$ with some non negligible constant which makes the approach essentially of theoretical interest. Nevertheless, the general idea is interesting and can be used in practice with some simplifications and this is the object of the present paper, which presents some *non asymptotycal* results based on the two following remarks :

1– Even if not negligible, the incidence graph of patches is still smaller than the incidence graph of original triangles, allowing to reduce the memory requirements.

2– $\frac{\log n}{12}$ is very small ($\frac{\log 70billions}{12} = 4$) it would be interesting to study in detail the composition of small fixed catalogs suitable for our purpose.

In the sequel we propose three catalogs and evaluate in details the amount of storage needed for representing triangulations using this approach. Preliminary implementation shows that the theoretical improvements are actually obtained in practice.

2 Definitions

A *Catalog* $\mathcal{C}$ is a collection $\{t_1, \ldots, t_p\}$ of planar triangulations with a simple boundary of arbitrary size, called *patches* (the t_i are called *tiny* triangulations in [6, 7]).

A *stable catalog* for a given operation is a catalog, where the result of each operation applied to any members of the catalog is either included in the catalog, or decomposable (within a restricted neighborhood) to some elements of the catalog. The interest of such a catalog is that a triangulation $\mathcal{T}$ could be represented with a disjoint union of patches in $\mathcal{C}$: $\mathcal{T} = \bigcup_{j\in J} t_{j_i}$ $(1 \leq j_i \leq p)$. Examples of stable catalogs are shown in Figure 1.

A catalog $\mathcal{C}$ is *minimal* if no patch t_i can be obtain as disjoint union of other patches in $\mathcal{C}$.

In this paper, we aim to support the insertion/deletion of degree 3 vertices and the edge flip as update operations.

Now one may ask, given a parameter k, how to find a minimal catalog $\mathcal{C}$ whose patches have each at least k triangles. We may proceed as follows:

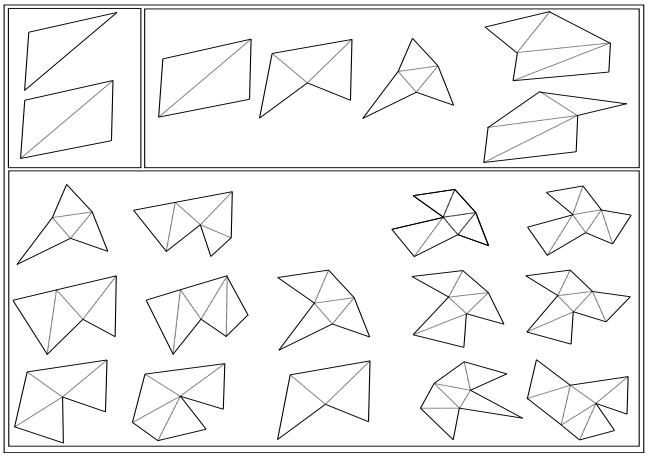

Figure 1: Three stable catalogs for the elementary operations : insertion and deletion of degree 3 vertex and edge flip. The first one is the Triangle-Quad catalog $\mathcal{C}_1$. $\mathcal{C}_2$ is a non-minimal catalog with at least 2 triangles/patch. $\mathcal{C}_3$ is minimal and contains at least 3 triangles/patch.

Firstly, the catalog should contains all of the patches with k triangles. Also, it includes all of the combinations with $2k-1$ triangles, since there is no way to represent a triangulation containing $2k-1$ triangles with only patches of k triangles. Then, we have to investigate all of the configurations produced when applying an update operation on the patches of the catalog. Whenever we obtain a configuration that is not decomposable into elements of the catalog $\mathcal{C}$, we add it to $\mathcal{C}$.

3 Simple Catalogs

In this section we present some simple catalogs, provided with upper bounds on the memory requirements of our structure, based on basic combinatorial assumptions.

3.1 The Triangle-quad catalog

The first catalog $\mathcal{C}_1$ is composed of two basic elements: triangles and quadrilaterals. It is clear that $\mathcal{C}_1$ is stable for the update operations considered above, but not minimal.

The triangulation is then represented by two sets, one storing the list of triangles, and one for the quads. The triangle representation remains unchanged: 6 references are required (3 for vertices and 3 for neighbors). For each quadrilateral only 4 references to vertices and 4 pointers to neighbors are needed, which makes save 2 references over each triangle converted into a quadrilateral.

The gain is then proportional to the number of constructed quadrilaterals. The maximum we can obtain

is $9n$ references for a triangulation of n points (for a quadrangulation). However, it is not always feasible to construct a triangulation only with quadrilaterals.

3.1.1 Static representation

In the static case we assume that the triangulation is entirely constructed: it only remains to get a partition into patches (as triangles and quads), as required by our scheme.

Several approaches has been proposed to convert triangulations to quadrangulations [9, 14], or create a quadrangulation from an input set of points [15]. In general, there is no obvious way to guarantee that an arbitrary triangulation could be converted to a quadrangulation. This is even impossible in some cases (when the size of the boundary is odd, see [5]). Otherwise, it is always possible to convert a triangulation of a topological sphere to a quadrangulation, since Petersen's theorem [12] guarantees that each 3-regular bridgeless connected graph has a complete matching (and hence all triangles can be gathered in quadrilaterals).

3.1.2 Dynamic representation

In the dynamic case we allow an incremental construction of the triangulation, provided with the implementation of basic update operations (vertex insertion/deletion and edge flip). Using Euler's formula, in the case of planar triangulations without boundary (triangulations of discrete surfaces homeomorphic to a sphere), we can state the following:

Lemma 1 *Let be $\mathcal{T}$ a planar triangulation with n vertices. Then an explicit Face-based representation using triangles and quads requires* $10.6n$ *pointers.*

Proof. We may assume we are given a decomposition of $\mathcal{T}$ such that there are not pairs of adjacent triangles (since adjacent triangles can be gathered to form a quadrilateral). Let t and q denote respectively the number of triangles and quadrilaterals in the decomposition of $\mathcal{T}$. The total number of triangles is

$$2n = t + 2q \tag{1}$$

The number of edges is known to be $3n$, and is equal to

$$3n = e_{int} + e_{ext} \tag{2}$$

where e_{int} is the number of edges internal to a quadrilateral (diagonals), and e_{ext} is the number of edges shared by quadrilaterals and triangles, and the other ones (between quadrilaterals). The first number is equal to q, while the second term is at least $3t$, since a triangle edge is always shared with a quad (triangles are not adjacent), and the number of quad should be positive, and hence :

$$3n - 3t - q > 0 \tag{3}$$

Using 1 and 3, we get :

$$q > \frac{3}{5}n \tag{4}$$

That means that we have at most $\frac{4}{5}n$ triangles. The triangulation can be represented with $\frac{53}{5}n = 10.6n$ references, instead of $13n$ in the basic representation, inducing a gain of 19%. □

3.2 Catalog with at least 2-triangles per package

We consider the catalog $\mathcal{C}_2$ with no less then 2 triangles in each patch (see Fig. 1): $\mathcal{C}_2$ is stable under our 3 updates operations but non minimal.

The quadrilaterals are represented as above. The pentagons are represented using 5 vertex references and 5 neighbor references. The hexagons are represented with 6 references for both vertices and neighbors. In this way, we can save two references in each triangle converted into quadrilateral, $\frac{8}{3}$ references in each triangle converted into pentagon, and 3 references when dealing with hexagons.

Again we may assume that no two quadrilaterals are adjacent. This is feasible since any two adjacent quadrilaterals can be converted into a hexagon. In this case (assuming that there is no hexagons in the worst case) we cannot get more than $\frac{5}{11}n$ quadrilaterals. On the other hand, we have at least $\frac{4}{11}n$ pentagons, which yields, in the worst case, a storing cost of $\frac{91}{11}n = 8.27n$, corresponding to a gain of 36% over the basic representation (using the same counting argument as before).

3.3 Minimal catalog with at least 3-triangles per package

Fig. 1 shows the minimal stable catalog $\mathcal{C}_3$ for the update operations with no less than three triangles per patch. This catalog contains triangulations having between 3 and 7 triangles, whose boundary has size between 6 and 9. Heptagons, octagons and enneagons are represented respectively with 14, 20 and 24 references. This produces respectively gains of $\frac{16}{5}$, $\frac{10}{3}$ and $\frac{24}{7}$ over each triangle converted into one of these patches.

The worst case cost for this catalog occurs when all the patches are pentagons. In this case, the global storage cost of the triangulation is $\frac{23}{3}n = 7.67n$ references, which is equivalent to 41% gain in memory space over the basic representation.

4 Implementation

Even thought we have implemented only the Triangle-Quad catalog $\mathcal{C}_1$, the following remarks hold for any arbitrary catalog. Firstly, the triangulation consists of multiple containers. One for the vertices data, and a container for each patch type in the catalog, which

20.000.000 points			
Number Points	Number Triangles	Number of Quads	Resident Memory
using quads	8.110.920	15.944.539	1.352.146.944
only triangles	39.999.998	-	1.691.283.456

Table 1: Memory requirements (expressed in byte units)using (1) a Quad/Triangle catalog, and (2) a basic representation. The gain is about 20%.

makes the references manipulated in the triangulation non equivalent. To avoid this problem, an additional information indicating the patch type has to be stored in the references. Another optional information to be stored, is a code concerning the pointed triangle in the patch. This can make faster the localization of a given triangle from a neighbor, avoiding checking all of the triangles in the patch. This issue is used in our implementation, since it takes insignificant size in our case(for triangle and quads one bit suffices).

5 Results

We have tested a 2D implementation of our package-based representation for triangulations using triangles and quadrialterals. The experimental results shown in Table 1 are obtained by computing the Delaunay triangulation, from uniform random distributions of points, adopting the incremental algorithm provided by CGAL [4]. It should be mentioned that our Quad-Triangle data structure can directly replace the usual CGAL data structure, without any modification of the end-user code. In term of software engineering, the use of catalogs to code triangulations requires some intermediate levels between the storage level (containing the real objects of the catalog that are elements of the catalog) and the user level (in which only logical faces appear). In our case, the Triangulation is a triple container : the first one is for vertices ; the second one is for triangles ; the third is for quadrilaterals. At higher level, the user manipulate *faces*, that are triangles, without worrying about the internal representation. These details affect the time processing of the triangulation in both steps : construction and manipulation.

6 Conclusion

We have presented a new approach for designing compact coding schemes for triangulations using stable catalogs. This scheme is inspired from previous theoretical work [6]. The idea induces some results concerning the bounds and the limits using some simple stable catalogs and promises a very efficient results if combined with other techniques. We have also implemented the smallest minimal catalog that includes triangles and quadrilaterals, and shown the practical bench results in term of memory requirements.

References

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In *Proc. 5th Workshop Algorithmic Found. Robotics)*, pages 83–96. A. K. Peters, 2001.

[2] P. Alliez and C. Gotsman. Advances in Multiresolution for Geometric Modelling. Recent Advances in Compression of 3D Meshes, 2005.

[3] D. K. Blandford, G. E. Blelloch, D. E. Cardoze and C. Kadow. Compact Representations of Simplicial Meshes in Two and ThreeDimensions. In *Proc. 12th International Meshing Roundtable, 2003.*

[4] J.-D. Boissonat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in CGAL *Comput. Geom. Theory & Applications* 22(13):519, May 2002

[5] P. Bose and G. Toussaint. Characterizing and efficiently computing quadrangulations of planar point sets. *Comput. Aided Geom. Des.*, 14(8):763-785,1997.

[6] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Succinct representation of triangulations with a boundary. In *Proc. of WADS 2005*, p. 134-145, 2005.

[7] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Optimal succinct representations of planar maps. In *Proc. SoCG*, 2006, to appear.

[8] The CGAL Manual" A. Fabri, E. Fogel, B. Gärtner, M. Hoffmann, L. Kettner, S. Pion, M. Teillaud, R. Veltkamp and M. Yvinec Release 3.0 , 2003.

[9] E. Heighway. A mesh generator for automatically subdividing irregular polygons into quadrilaterals. In *IEEE Transactions on Magnetics*, , 19(6):2535–2538, 1983.

[10] M. Kallmann and D. Thalmann Star-vertices: a compact representation for planar meshes with adjacency information *J. Graph. Tools*, 6(1):7-18, 2001.

[11] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry – Theory and Applications*, 13:65–90, 1999.

[12] J. Petersen. Die Theorie der regularen Graphs (The theory of regular graphs). Acta Mathematica, 15:193–220, 1891.

[13] Dominique Poulalhon and Gilles Schaeffer. Optimal coding and sampling of triangulations. In *Proc. of ICALP*, p. 1080-1094, 2003.

[14] S. Ramaswami, P.A. Ramos and G.T. Toussaint. Converting triangulations to quadrangulations. *Comput. Geom. Theory & Applications* 9(4):257-276, 1998.

[15] G. Toussaint. Quadrangulations of Planar Sets. In *Proc. of WADS*, 218–227, 1995.